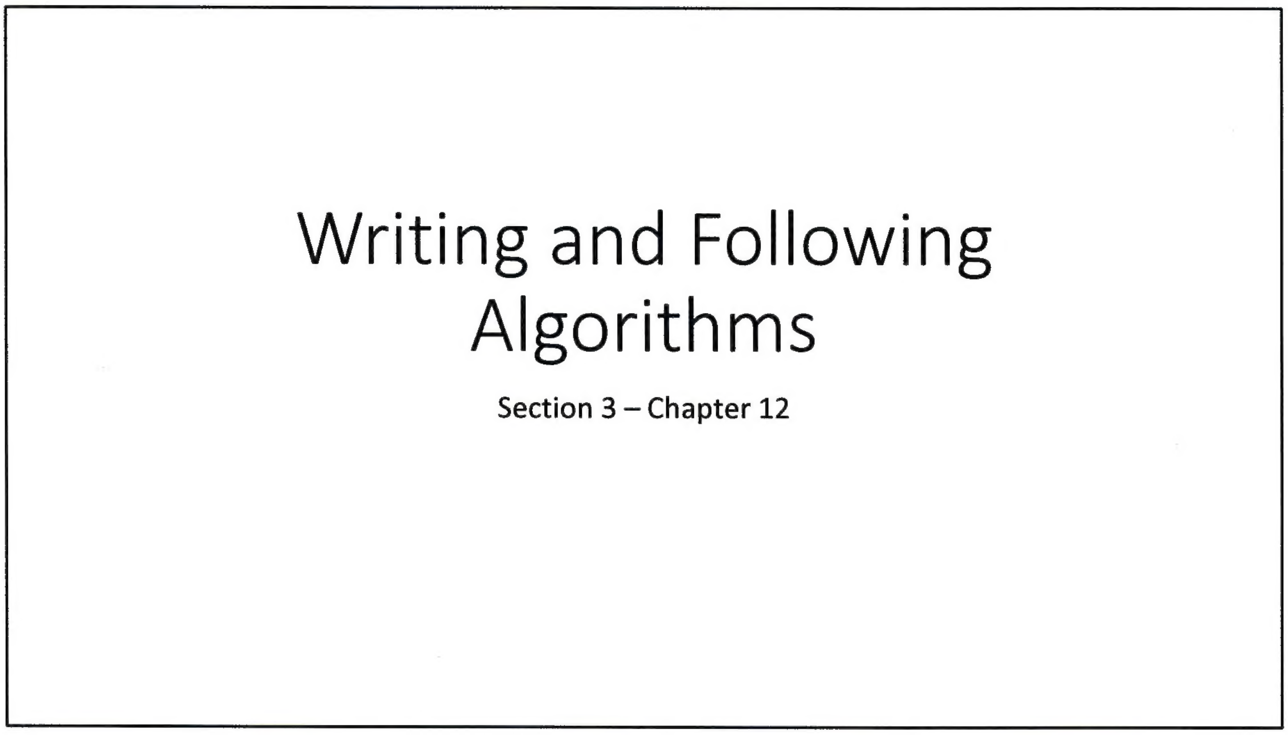
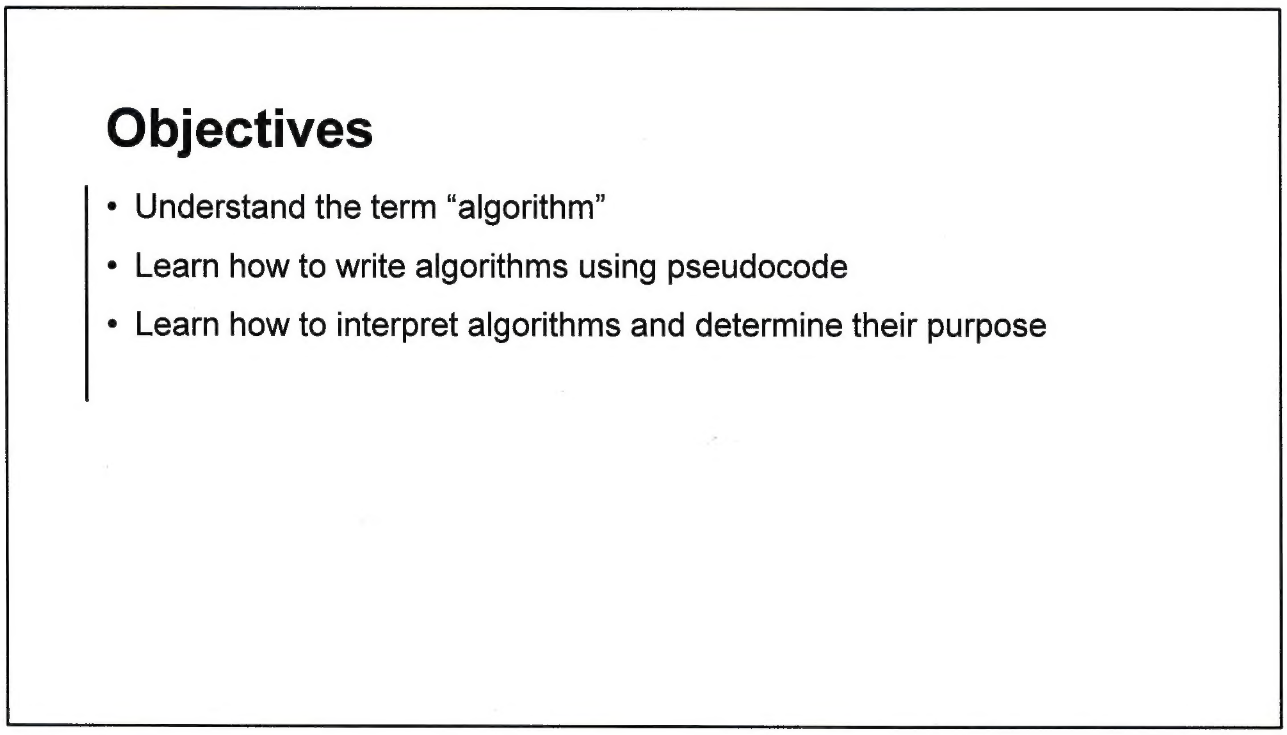


Writing and Following Algorithms

Section 3 – Chapter 12



Objectives

- Understand the term “algorithm”
- Learn how to write algorithms using pseudocode
- Learn how to interpret algorithms and determine their purpose

Key Words:

- Algorithm –
    - A set of instructions to solve a problem or complete some well-defined task in a finite number of steps.
- Bubble Sort –
    - Each item is compared with the adjacent item and swapped if it is larger.
- Trace Table –
    - Follow through an algorithm to determine its purpose or to find a logic error.

[illegible]ing and following algorithms [illegible]

Connection

Algorithm

- An algorithm is a set of instructions to solve a problem or complete some well-defined task in a finite number of steps
- A recipe for chocolate cake, or a knitting pattern, are algorithms of a kind

Activation

Computational algorithms

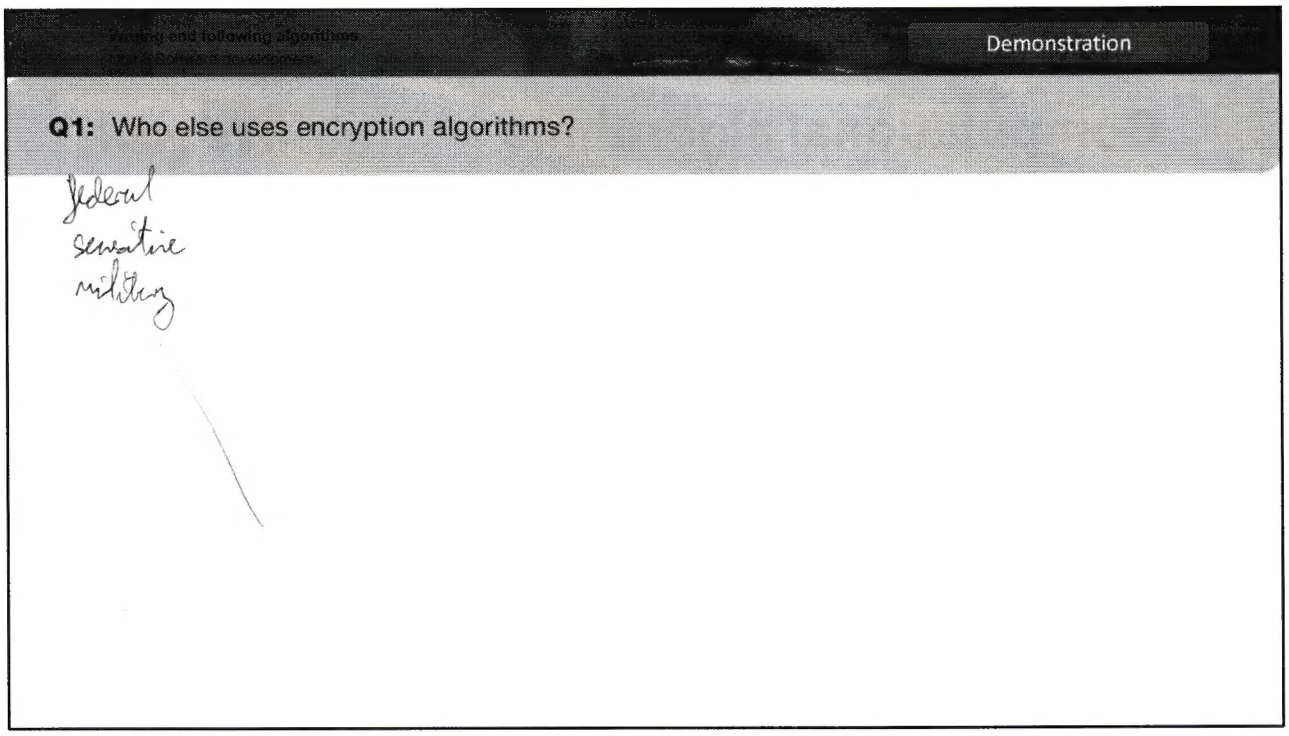
- Give some examples of computational algorithms

There are thousands of real-world problems
to be solved using different algorithms;
some of them still unsolved!

Activation

Problems solved by algorithms

- **Routing** problems:
  - Routing packets of data around the Internet by the shortest route
  - Finding the shortest route for a salesman to cover his territory
- **Timetabling** commercial aircraft crews so that they do not exceed their permitted flight hours
- **Searching** information on the Internet or from a database
- **Encrypting** communications so that they cannot be hacked
- **Sorting** large amounts of data
- **Writing a compiler** program to translate a high level language to machine code

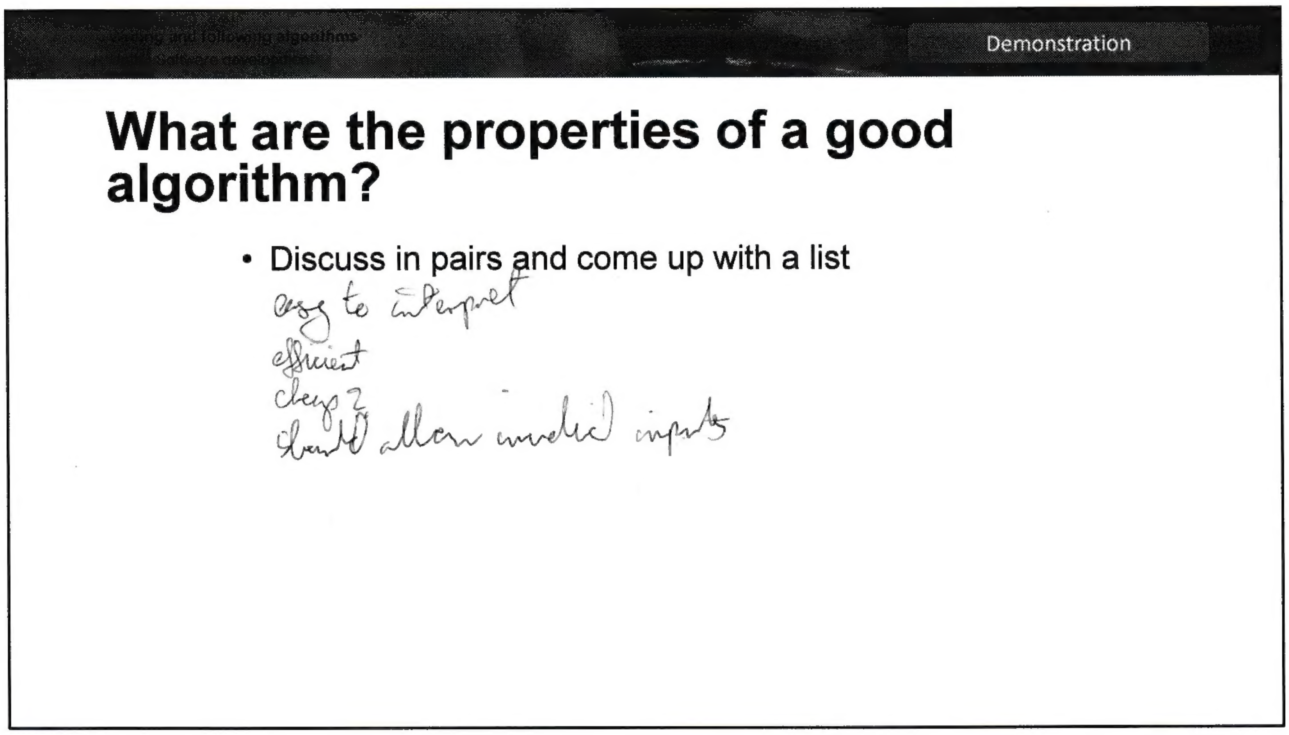


Writing and following algorithms
Software development

Demonstration

Q1: Who else uses encryption algorithms?

federal
sensitive
military



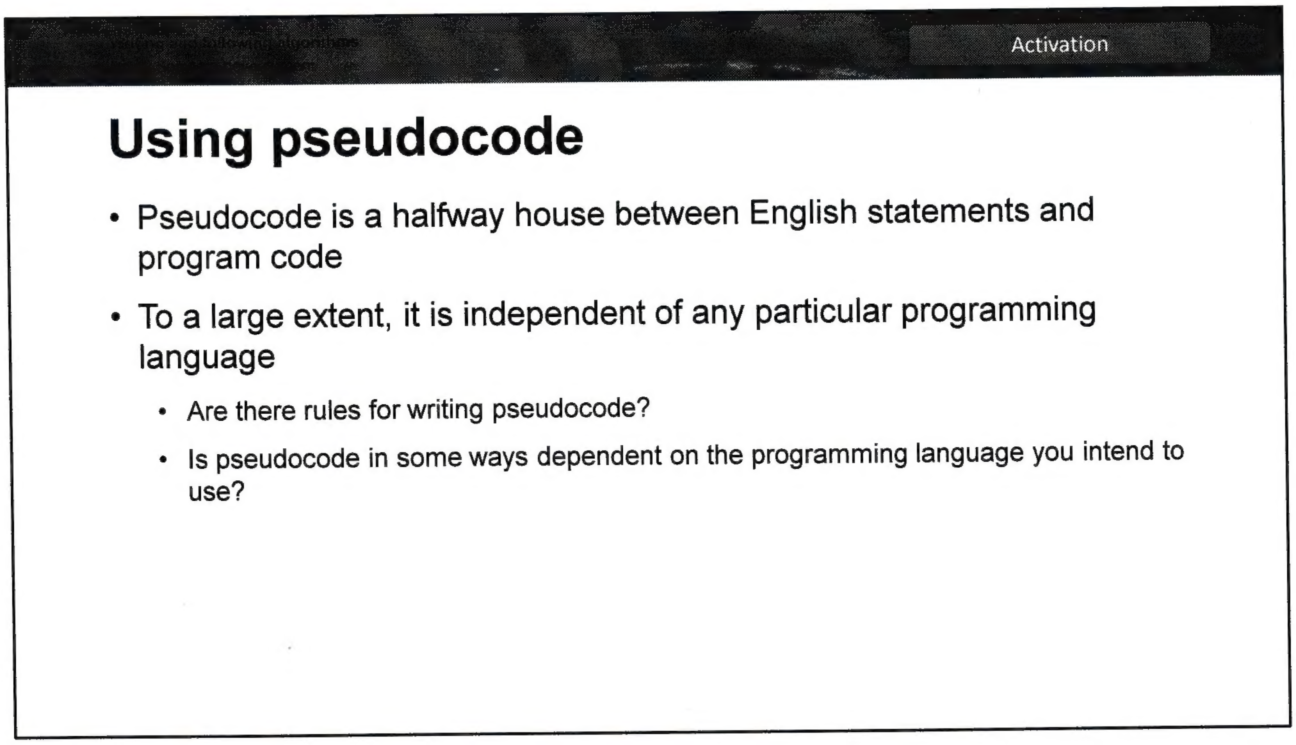
Writing and following algorithms
Software development

Demonstration

What are the properties of a good algorithm?

- Discuss in pairs and come up with a list

easy to interpret
efficient
cheap?
shouldn't allow invalid inputs

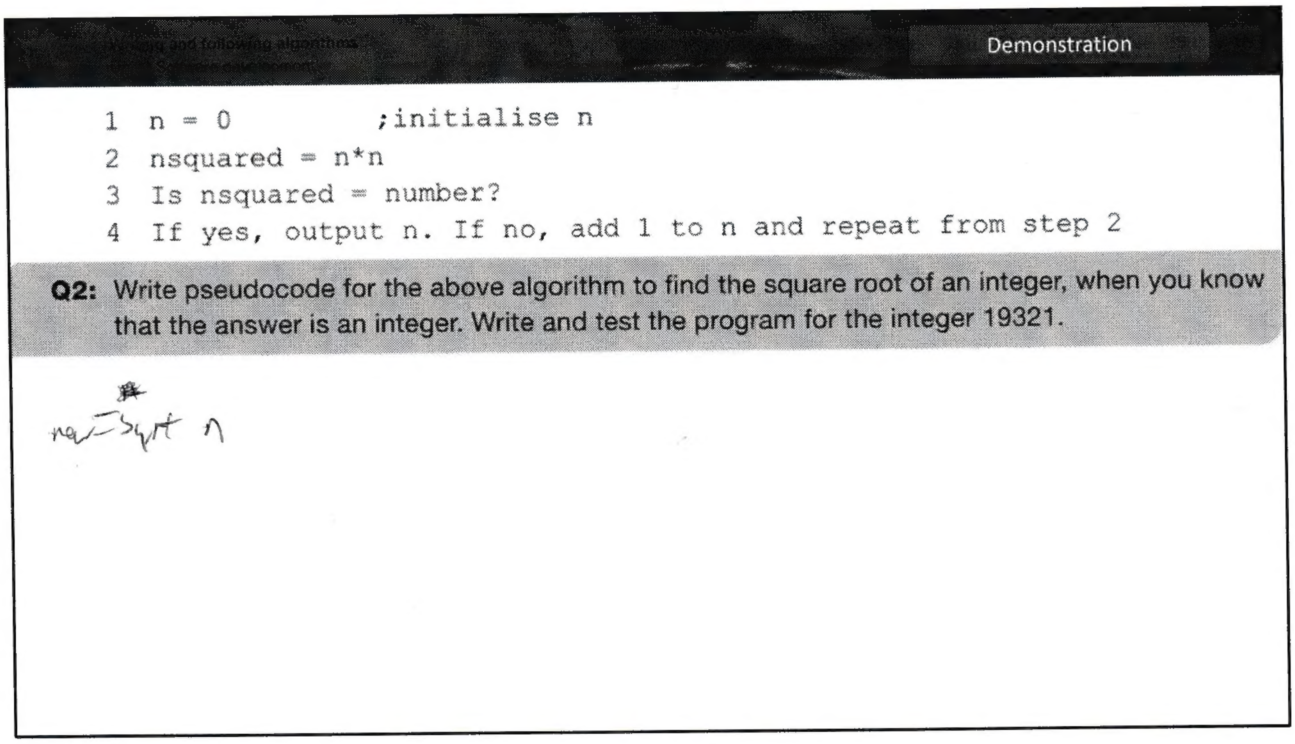


Writing and following algorithms

Activation

Using pseudocode

- Pseudocode is a halfway house between English statements and program code
- To a large extent, it is independent of any particular programming language
    - Are there rules for writing pseudocode?
    - Is pseudocode in some ways dependent on the programming language you intend to use?



Writing and following algorithms

Demonstration

```
1  n = 0          ;initialise n
2  nsquared = n*n
3  Is nsquared = number?
4  If yes, output n. If no, add 1 to n and repeat from step 2
```

Q2: Write pseudocode for the above algorithm to find the square root of an integer, when you know that the answer is an integer. Write and test the program for the integer 19321.

new = sqrt n

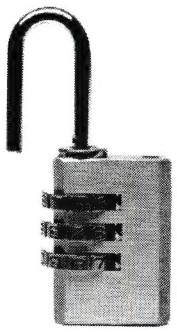
Demonstration

Demonstration

Q3: Which of the properties of a good algorithm, stated at the start of the chapter, does this algorithm not satisfy?

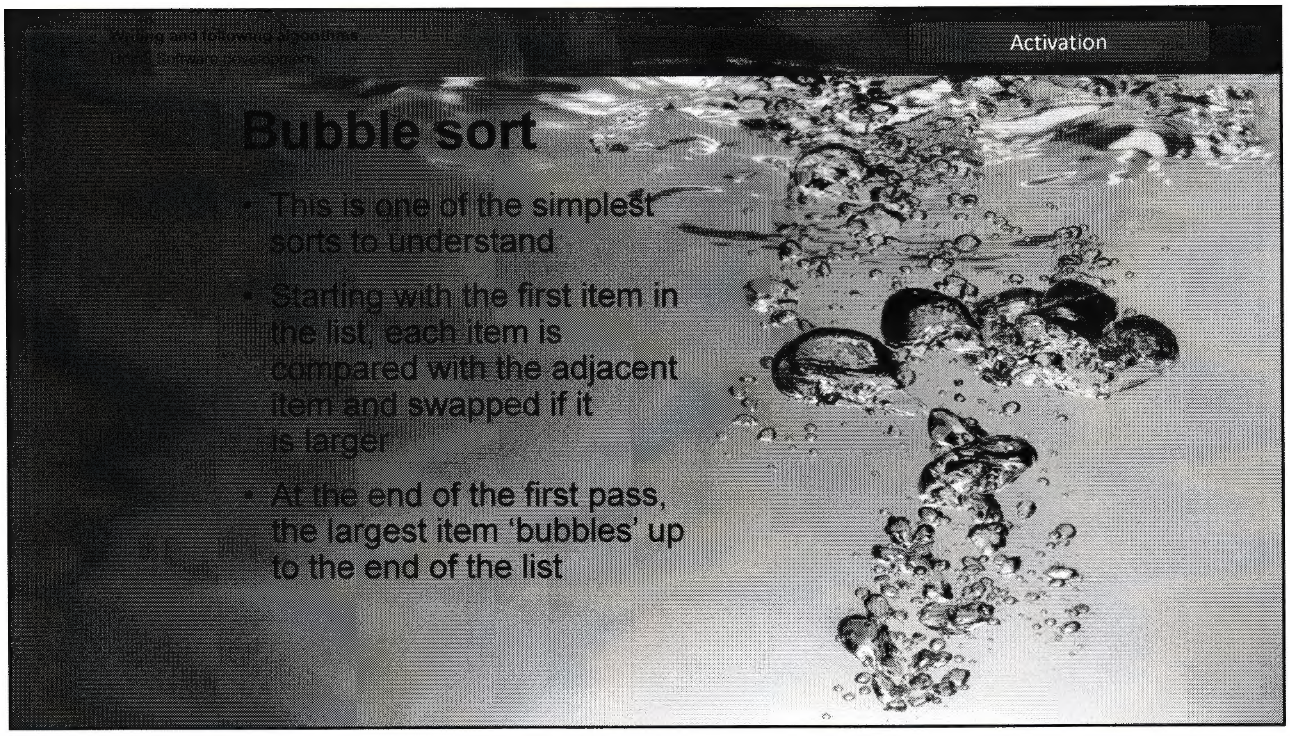
Worksheet 2

- Try **Task 1** on **Worksheet 2**



Sorting algorithms

- There are many different sorting algorithms, from very simple ones which execute very slowly, (e.g. an insertion sort) to more complex ones which execute very fast (e.g. merge sort)
- On a given computer, a fast sort algorithm may sort ten million numbers in say 20 minutes, whereas an inefficient algorithm may take more than a month to sort the same data

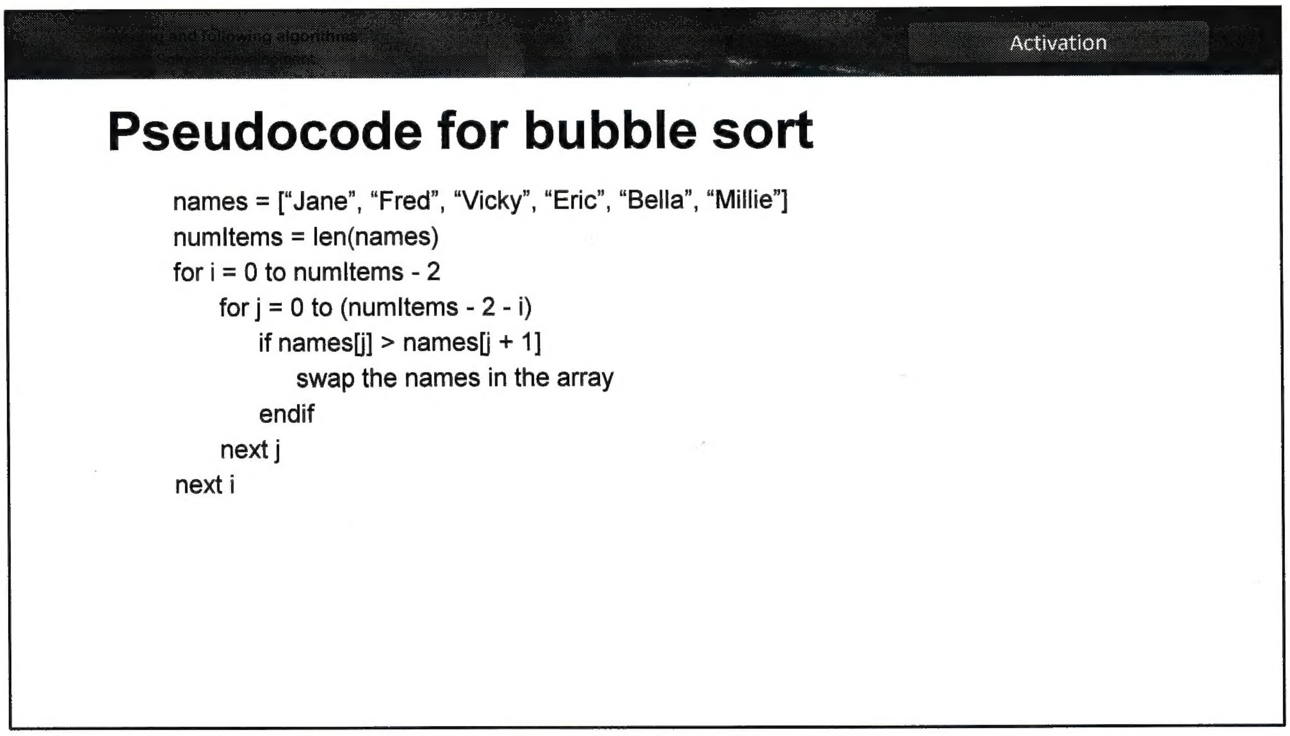


Writing and following algorithms
GCSE Software development

Activation

Bubble sort

- This is one of the simplest sorts to understand
- Starting with the first item in the list, each item is compared with the adjacent item and swapped if it is larger
- At the end of the first pass, the largest item 'bubbles' up to the end of the list



Writing and following algorithms
GCSE Software development

Activation

Pseudocode for bubble sort

```
names = [“Jane”, “Fred”, “Vicky”, “Eric”, “Bella”, “Millie”]
numItems = len(names)
for i = 0 to numItems - 2
    for j = 0 to (numItems - 2 - i)
        if names[j] > names[j + 1]
            swap the names in the array
        endif
    next j
next i
```

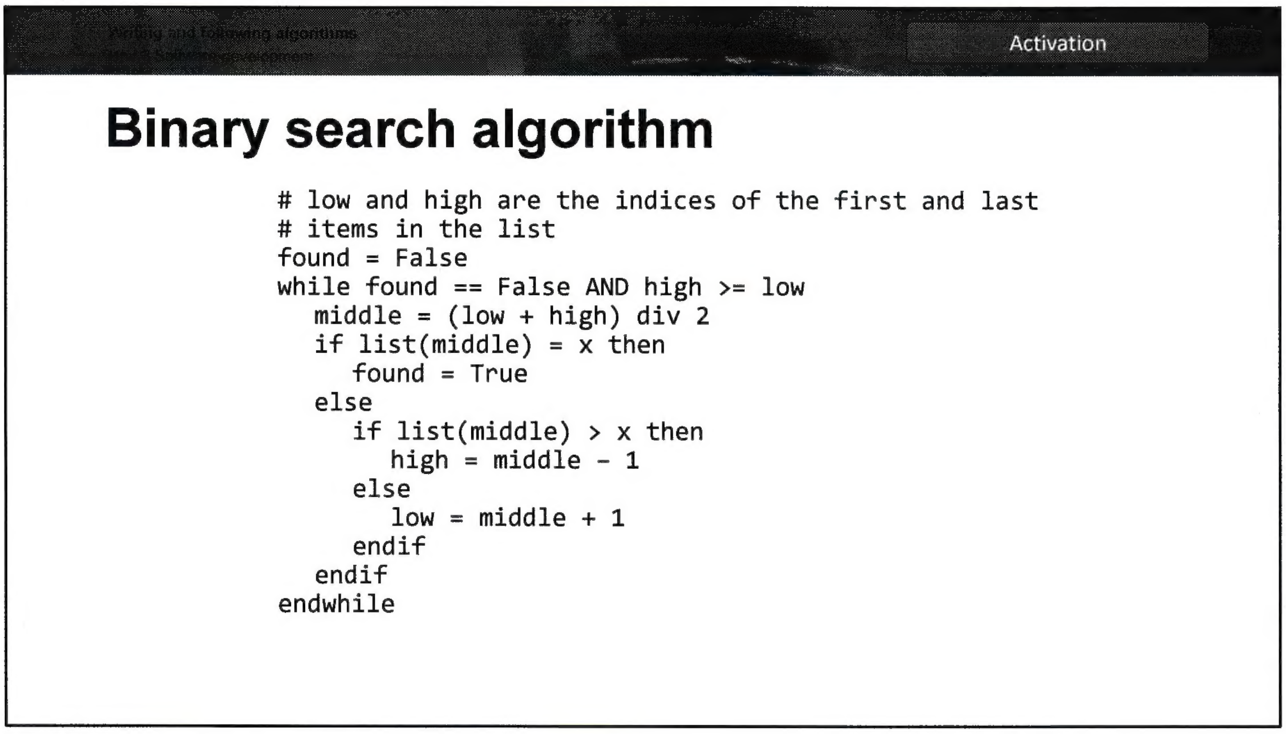
Bubble sort

- Suppose you have a list to be sorted alphabetically

  names = ["Henry", "Steve", "Julie", "Ava", "Tom", "Olivia"]
- After one pass though the bubble sort algorithm, what order will the names be in?
- How many passes will be needed to sort the list into ascending sequence?
- Write an algorithm to swap two items names[j] and names[j + 1]

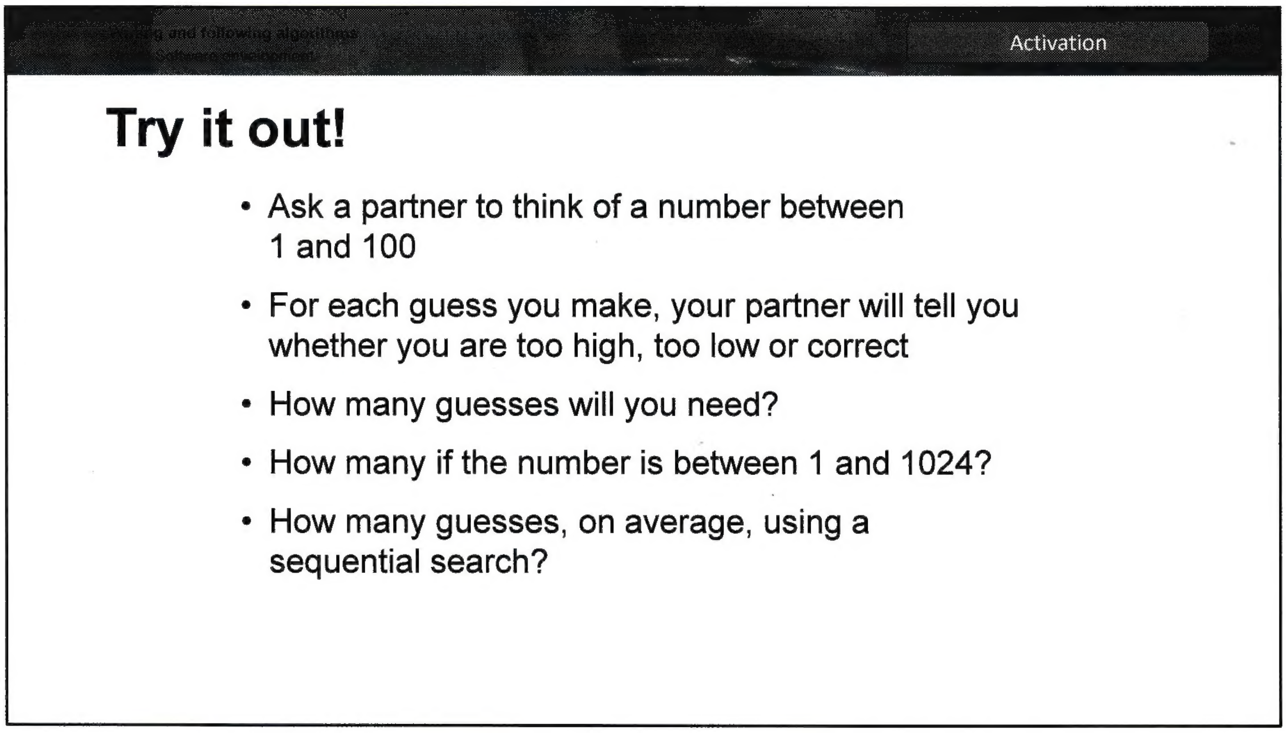
Searching algorithms

- Two of the most common searching algorithms are the linear search and the binary search
- The linear search starts at the beginning of the list and examines every item
- The binary search uses the "divide and conquer" strategy to halve the search area each time an item is examined
  - It can only be used if the items are in sorted order – they could be alphabetic or numeric, for example


Activation

Binary search algorithm

```
# low and high are the indices of the first and last
# items in the list
found = False
while found == False AND high >= low
   middle = (low + high) div 2
   if list(middle) = x then
      found = True
   else
      if list(middle) > x then
         high = middle - 1
      else
         low = middle + 1
      endif
   endif
endwhile
```


Activation

Try it out!

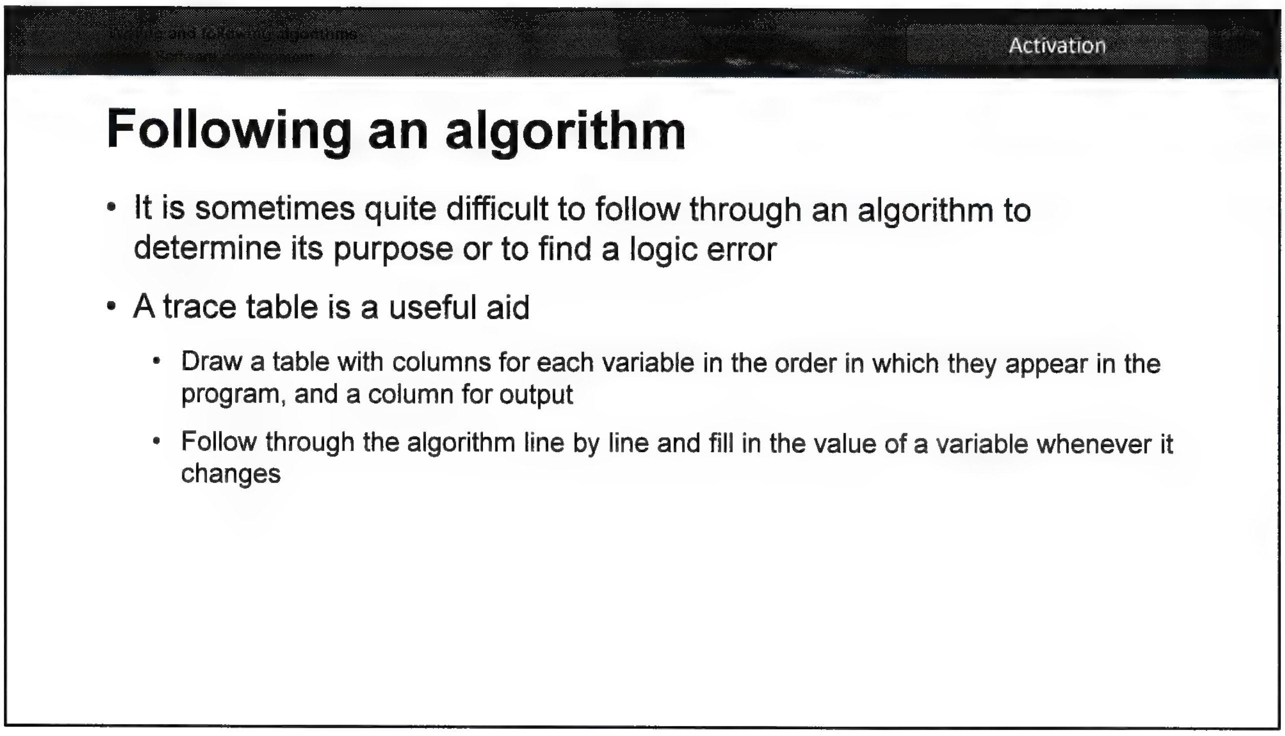
- Ask a partner to think of a number between 1 and 100
- For each guess you make, your partner will tell you whether you are too high, too low or correct
- How many guesses will you need?
- How many if the number is between 1 and 1024?
- How many guesses, on average, using a sequential search?

Evaluating a program

- You have seen that some algorithms are much more efficient than others at solving a problem
- What about program code?
- What makes one program "better" than another?
  - Suggest ways of making your programs look like the work of a professional!

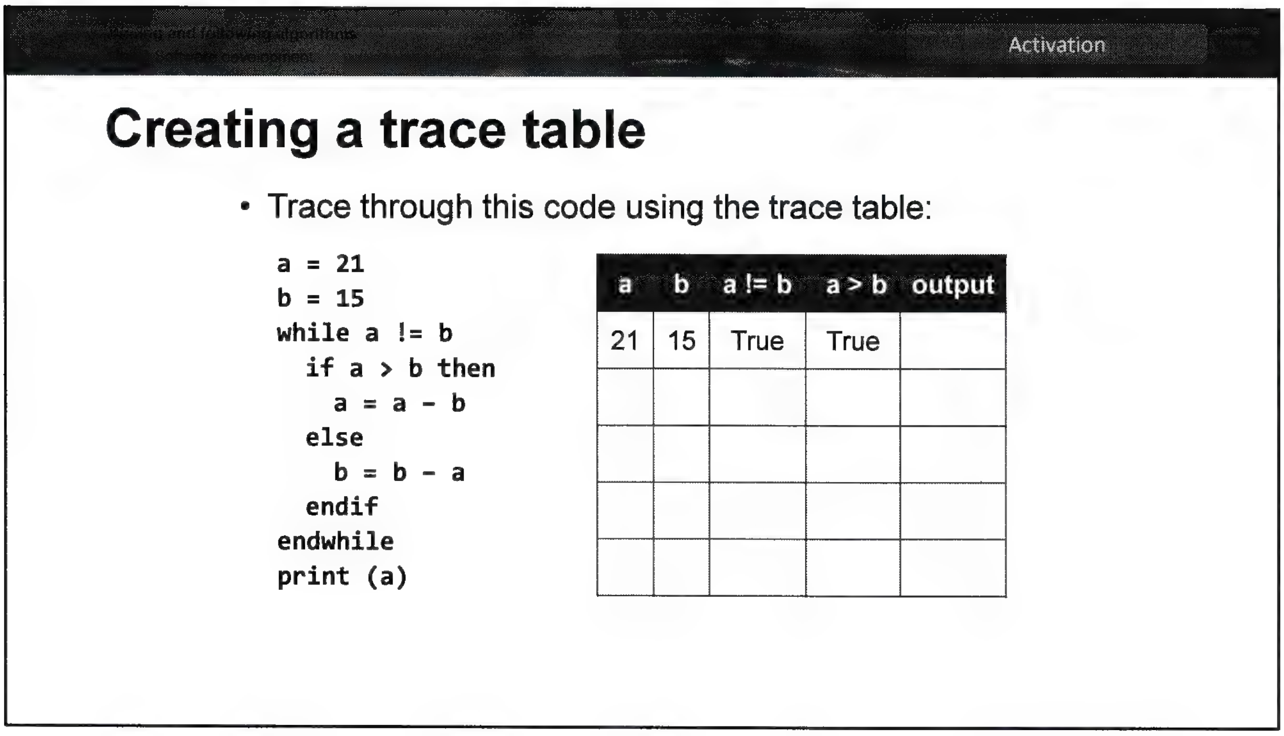
Writing "good" programs

- Use comments to document your programs
- Use a standard for identifiers – e.g.
  - start all variable names with a lowercase letter
  - use CamelCaps rather than spaces or underscores
- Use properly indented code
- Make sure your program does not contain statements that are not needed
- Use a modular structure, with no module longer than a page of code, performing a discrete task



Following an algorithm

- It is sometimes quite difficult to follow through an algorithm to determine its purpose or to find a logic error
- A trace table is a useful aid
  - Draw a table with columns for each variable in the order in which they appear in the program, and a column for output
  - Follow through the algorithm line by line and fill in the value of a variable whenever it changes



Creating a trace table

- Trace through this code using the trace table:

```
a = 21
b = 15
while a != b
  if a > b then
    a = a - b
  else
    b = b - a
  endif
endwhile
print (a)
```

a	b	a != b	a > b	output
21	15	True	True	

Creating a trace table

- Trace through this code using the trace table:

```
a = 21
b = 15
while a != b
  if a > b then
    a = a - b
  else
    b = b - a
  endif
endwhile
print (a)
```

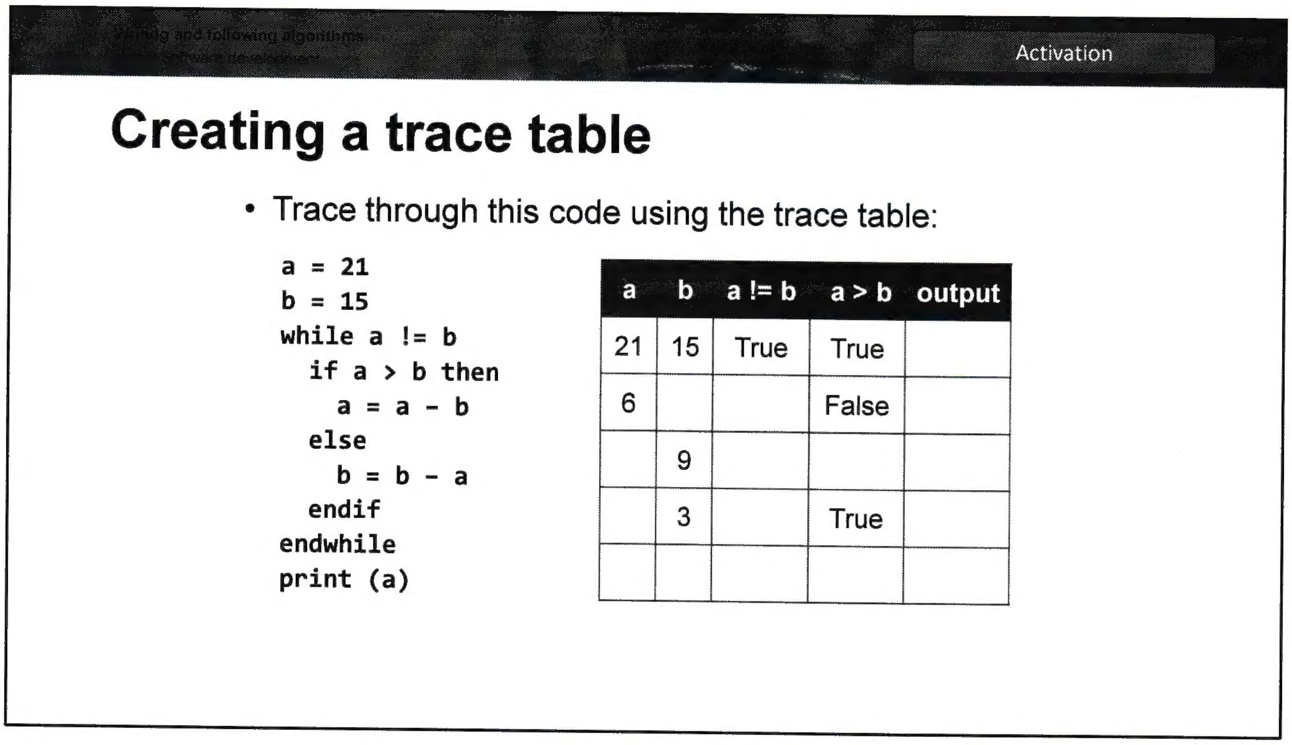
a	b	a != b	a > b	output
21	15	True	True	
6			False	

Creating a trace table

- Trace through this code using the trace table:

```
a = 21
b = 15
while a != b
  if a > b then
    a = a - b
  else
    b = b - a
  endif
endwhile
print (a)
```

a	b	a != b	a > b	output
21	15	True	True	
6			False	
	9			

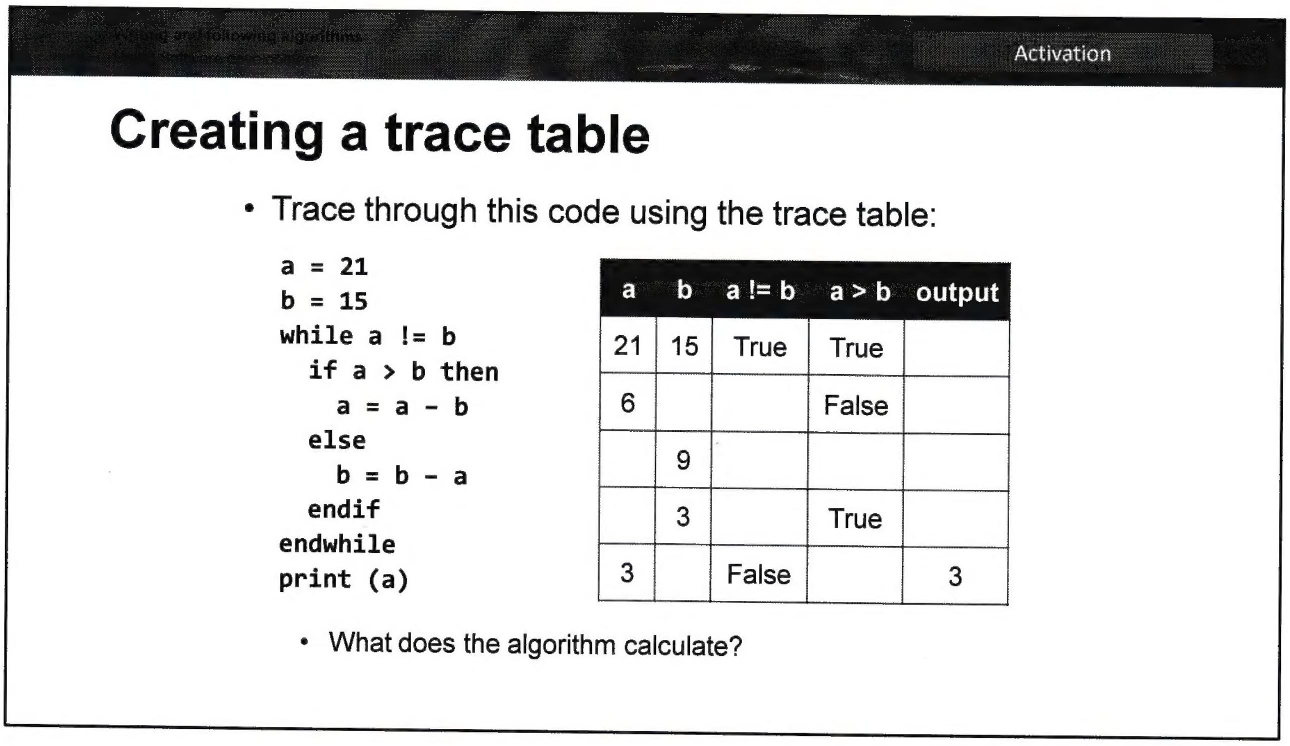


Creating a trace table

- Trace through this code using the trace table:

```
a = 21
b = 15
while a != b
  if a > b then
    a = a - b
  else
    b = b - a
  endif
endwhile
print (a)
```

a	b	a != b	a > b	output
21	15	True	True	
6			False	
	9			
	3		True	



Creating a trace table

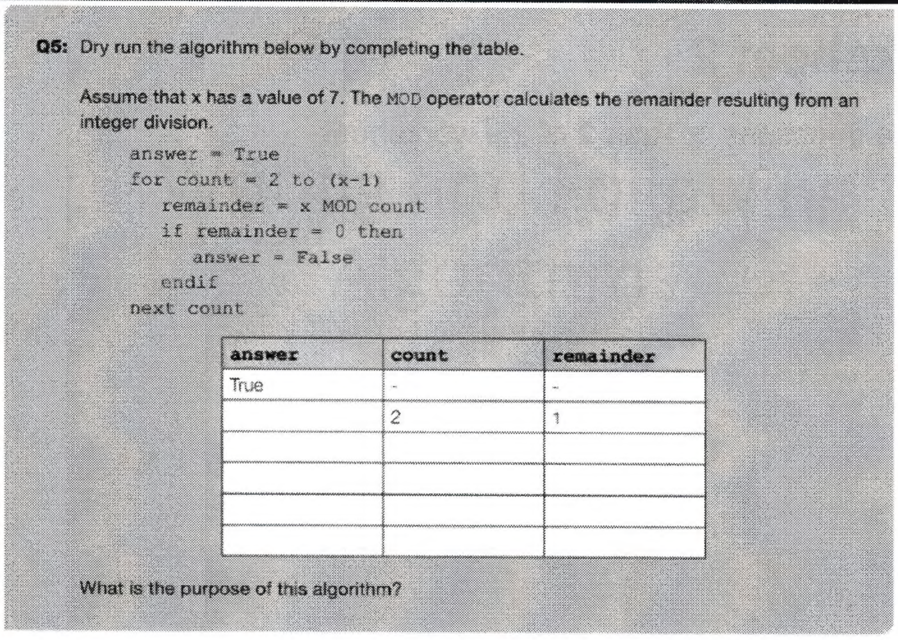
- Trace through this code using the trace table:

```
a = 21
b = 15
while a != b
  if a > b then
    a = a - b
  else
    b = b - a
  endif
endwhile
print (a)
```

a	b	a != b	a > b	output
21	15	True	True	
6			False	
	9			
	3		True	
3		False		3

- What does the algorithm calculate?

Demonstration



Q5: Dry run the algorithm below by completing the table.

Assume that x has a value of 7. The `MOD` operator calculates the remainder resulting from an integer division.

```
answer = True
for count = 2 to (x-1)
   remainder = x MOD count
   if remainder = 0 then
      answer = False
   endif
next count
```

answer	count	remainder
True	-	-
	2	1

What is the purpose of this algorithm?

Demonstration

What is the purpose of this algorithm?

Worksheet 2

- Try the questions in **Task 2** of the worksheet

Consolidation

- Pseudocode is useful for designing algorithms
- Sorting and searching and two very common operations in data processing, and efficient algorithms can greatly reduce execution times
- Trace tables are useful for following through algorithms
- Designing and tracing through algorithms are two vital skills for computer scientists